Informe sobre creación y configuración de una aplicación de agenda que permita insertar, editar, eliminar y leer registros utilizando Firebase incluyendo el desarrollo de un icono personalizado y un splash screen

Xavier Calle, Sebastián Morales
Escuela de Formación de Tecnólogos
Escuela Politécnica Nacional
xavier.calle@epn.edu.ec, sebastian.morales@epn.edu.ec.

Resumen-
***En este documento se explica como crear una aplicación CRUD utilizando la base de datos de Firebase, Real time data base para ser mas exactos*. Esta es una aplicación creada en Ionic de tipo angular, se utiliza Cordova que es un marco de desarrollo móvil de código abierto, y así mismo se utilizan las respectivas librerías y paquetes de Firebase para angular. Se realiza la respectiva configuración de cada archivo y se explica para que sirven ciertas cosas en el código, para obtener como resultado una aplicación CRUD utilizando Firebase, finalmente se muestran las capturas de pantalla de los módulos creados dentro de la aplicación logrando así cumplir el objetivo propuesto.**

I. Introducción

Muchas de las veces para desarrollar aplicaciones para dispositivos móviles es necesario aprender lenguajes nativos de programación para cada plataforma y esto dificulta su desarrollo, Ionic nos permite programar de manera "sencilla" aplicaciones multiplataforma, ya que utiliza lenguajes de programación web para el desarrollo de proyectos que posteriormente podremos instalar en diversas plataformas.

II. DESARROLLO

A. *Conexión a Firebase.*

Es necesario crear un proyecto en Firebase.

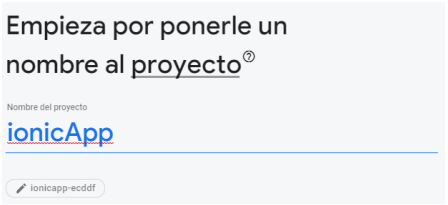

Fig. 1. Creación del proyecto en Firebase.

Registramos una aplicación.

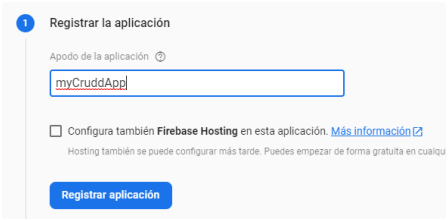

Fig. 2. Registro de la aplicación en Firebase.

Copiamos la variable de configuración de nuestro proyecto y aplicación en Firebase.

B. *Creación de la aplicación.*

Primeramente, se debe ubicar donde se alojará el proyecto dentro del sistema, conociendo la ubicación se abre una consola de comandos y se ejecuta el comando "ionic start nombre-proyecto blank --type=angular".

Posterior a ello dentro de la carpeta del proyecto se ejecuta el comando "npm install firebase @anguar/fire –save" este comando lo que hace es instalar el paquete de firebase y @angular/fire lo que implica que también se instalaran las librerías necesarias para el proyecto.

C. *Estructura del proyecto*

La carpeta que mas interesa en este caso es la "src" dentro de esta carpeta se encuentra a carpeta "app" y dentro de esta se encuentra una carpeta llamada "home" donde se encontraran varios archivos, los dos archivos principales que interesan son "home.page.html" y "home.page.ts". El archivo de tipo HTML va a proporcionar la parte visual, mientras que el archivo TS va a proporcionar la lógica.

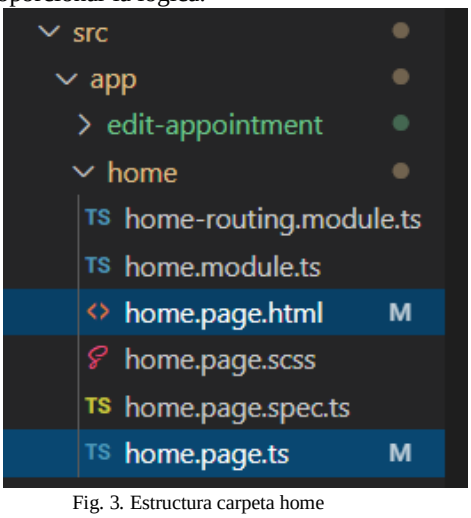

Fig. 3. Estructura carpeta home

D. *Configuración de variables de entorno.*

Dentro de la carpeta “src” se encuentra la carpeta “enviroments” donde se encontrarán dos archivos .ts en los cuales se va a configurar la cadena de conexión que nos provee Firebase como se muestra en la figura 3 y los archivos deben quedar como se muestran en la figura 4.

La diferencia de los dos archivos que se encuentran en esta carpeta es que uno se utiliza para la producción y el otro no.

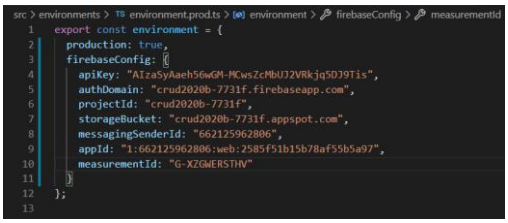

Fig. 4. Cadena de conexión a Firebase

E. Implementación de librerías de firebase.

Es necesario importar las librerías e inicializarlas para utilizarlas en el proyecto como el acceso a la base de datos, a la autenticación y al storage, esto se realiza en el archivo “app.module.ts” que se encuentra dentro de la carpeta “src”.

```
//firebase librerías
import {AngularFireModule} from '@angular/fire';
import {AngularFireAuthModule} from '@angular/fire/auth';
import {AngularFireDatabaseModule} from '@angular/fire/database';
import {AngularFireStorageModule} from '@angular/fire/storage';

//environmnet
import {environment} from '../environments/environment';
```

Fig. 5. Importación de librerías de Firebase y enviroment.

Dentro del archivo “app.module” se posee una función a la que se le atribuye la creación del proceso inicial de un módulo, esta función es @NgModule en la cual nos permite cargar elementos que se vayan a utilizar cuando se inicialice la aplicación y así mismo se pueden cargar elementos en específico para ciertos módulos, es decir, otras pantallas.

```
AngularFireModule.initializeApp(environment.firebaseConfig),
AngularFireAuthModule,
AngularFireDatabaseModule,
AngularFireStorageModule
```

Fig. 6. Inicialización de variables.

F. Generar módulos necesarios.

Para esto se utiliza el comando “ionic generate page make-appointment” en la consola de comandos dentro de la carpeta del proyecto, este comando genera una estructura de carpetas con los archivos y el contenido necesarios. En estos archivos se configurará la lógica y la parte visual para poder generar un contacto.

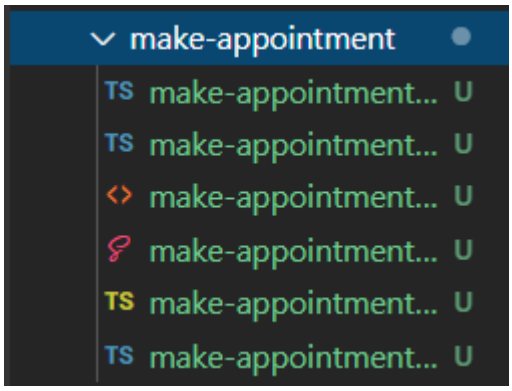

Fig. 7. Estructura de archivos carpeta make-appointment

Así mismo se utiliza el comando “ionic generate page edit-appointment” el cual se va a configurar tanto visualmente como de manera lógica la edición de un contacto.

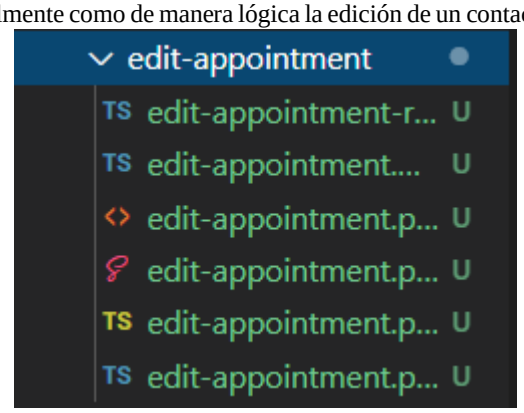

Fig. 8. Estructura de archivos carpeta edit-appointment

Dentro de la carpeta “app” que se encuentra en la carpeta “src” se encuentra un archivo llamado “app-routing.module” en el cual se habrá realizado un enrutamiento respectivo a los diferentes módulos que se crearon anteriormente.

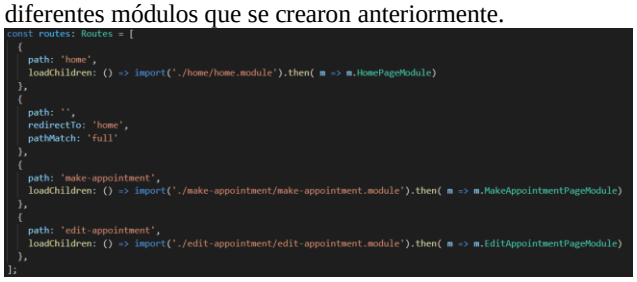

Fig. 9. Ruteo de los módulos agregados

G. Generar un servicio.

Para esto se utiliza el comando “ionic service shares/appointment” donde se creará un archivo llamado “Appointment .ts” el cual va a poseer una clase con los elementos que se van a aguardar en la base de datos.

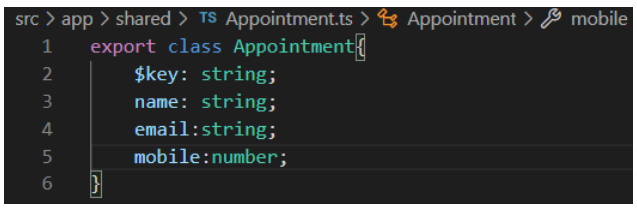

Fig. 10. Appointment.js

Y en el archivo llamado “appointment.service.ts” se ponen los servicios generales de la aplicación es decir los métodos CRUD, se importan las librerías necesarias para poder trabajar y también el archivo creado (Appointemnt.ts)

```
// Create
createBooking(apt: Appointment) {
  return this.bookingListRef.push({
    name: apt.name,
    email: apt.email,
    mobile: apt.mobile
  })
}

// Get Single
getBooking(id: string) {
  this.bookingRef = this.db.object('/appointment/' + id);
  return this.bookingRef;
}

// Get List
getBookingList() {
  this.bookingListRef = this.db.list('/appointment');
  return this.bookingListRef;
}

// Update
updateBooking(id, apt: Appointment) {
  return this.bookingRef.update({
    name: apt.name,
    email: apt.email,
    mobile: apt.mobile
  })
}

// Delete
deleteBooking(id: string) {
  this.bookingRef = this.db.object('/appointment/' + id);
  this.bookingRef.remove();
}
}
```

Fig. 11. Métodos CRUD (appointment.service.ts)

H. Configuración de archivos TS y HTML.

Los archivos como make-appointment.ts y edit-appointemt.ts poseen la función ngOninit () la cual inicializa un componente y así mismo se pueden crear muchas más funciones como por ejemplo se crea una para validar que los datos que se ingresan al agendar a un contacto no estén vacíos.

Por otro lado, los archivos HTML pueden ser configurados como se desee, en este caso se colocan tres inputs para la inserción de datos y un botón de tipo submit para poder guardar los datos en Firebase de igual modo la pantalla de editar, solo que en esta aparecerán los datos cuando se seleccione un contacto.

En la página de home.html se encuentran todos los elementos que se muestran en la pantalla principal por lo que gracias al uso de *ngFor para consultar la base de datos y mostrar un listado de todos los contactos que se encuentran guardadas en la base de datos y así mismo se colocan 2 iconos a lado de cada cita el primero para editar y el segundo para eliminar el contacto.

```
<ion-content>
  <ion-list class="ios list-ios hydrated">
    <ion-list-header class="ios hydrated">
      Appointments
    </ion-list-header>

    <ion-item *ngFor="let booking of Bookings" class="user-list">
      <ion-label>
        <h5>
          <ion-icon name="person"></ion-icon> {{booking.name}}
        </h5>
        <p>
          <ion-icon name="call"></ion-icon> {{booking.mobile}}
        </p>
      </ion-label>

      <div class="item-note" item-end>
        <button ion-button clear [routerLink]="['/edit-appointment/', booking.$key]">
          <ion-icon name="create" style="zoom:2.0"></ion-icon>
        </button>
        <button ion-button clear (click)="deleteBooking(booking.$key)">
          <ion-icon name="trash" style="zoom:2.0"></ion-icon>
        </button>
      </div>
    </ion-item>
  </ion-list>
```

Fig. 12. Estructura página home.html

I. Splash screen e ícono.

Necesitamos tener una imagen y un ícono que cumplan con los límites que nos otorga Cordova para el manejo de iconos y splash screens. Reemplazamos el icono y splash screen que viene por defecto en la carpeta de recursos del proyecto.

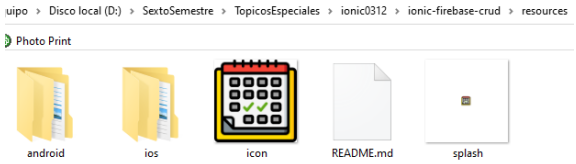

Fig. 13. Carpeta de recursos del proyecto.

Una vez realizado el cambio corremos el comando ionic cordova resources Cordova configura con este comando el ícono de la aplicación y el splash screen de la misma.

```
D:\SextoSemestre\TopicosEspeciales\ionic0312\ionic-firebase-crud\resources>ionic cordova resources
> cordova-res.cmd
[cordova-res] Generated 18 resources for Android
[cordova-res] WARN:     Source icon "resources/icon.png" contains alpha channel, generated icons for iOS will not.
[cordova-res]
[cordova-res]   Apple recommends avoiding transparency. See the App Icon Human Interface Guidelines[1] for details. Any
transparency in your icon will be filled in with white.
[cordova-res]
[cordova-res]   [1]: https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/
[cordova-res]
[cordova-res] Generated 47 resources for iOS
[cordova-res] Wrote to config.xml

D:\SextoSemestre\TopicosEspeciales\ionic0312\ionic-firebase-crud\resources>
```

Fig. 14. Comando de cordova para configurar el icono y el splash screen.

J. Capturas de la aplicación

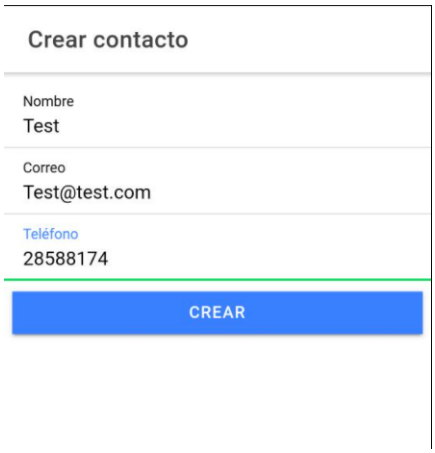

Fig. 13. Página de creación de contacto

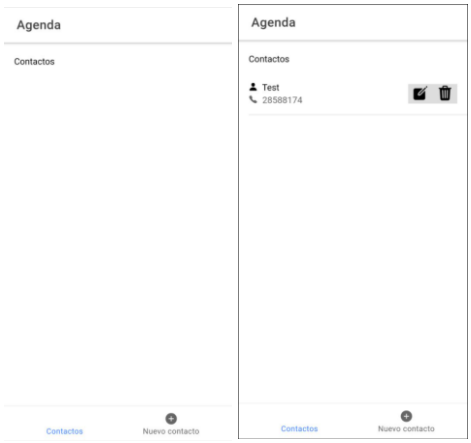

Fig. 14. Pagina de home vacía y con contactos

Fig. 15. Página de edición del contacto.

II. CONCLUSIONES

Ionic nos facilita crear una aplicación móvil a partir de lenguajes que originalmente se utilizan para aplicaciones web.

Cordova permite utilizar tecnologías web como HTML5, CSS y JavaScript para poder desarrollar en multiplataforma con el objetivo de evitar el lenguaje de desarrollo nativo de cada plataforma móvil.

Integrar una aplicación de firebase a nuestro proyecto es muy sencillo ya que solo necesitamos configurar la cadena de conexión de nuestra aplicación e importar las librerías necesarias.

Es necesario tener un ícono y un splash screen que cumplan con los tamaños mínimos para utilizar cordova resources e integrarlos al proyecto.